The Reactive Accompanist: Adaptation and Behavior Decomposition in a Music System

Joanna Bryson

The Artificial Intelligence Laboratory, MIT NE43-825, Cambridge, MA 02139, USA
joanna@ai.mit.edu

Abstract This paper describes a reactive, behavior-based system that mimics a human musical skill — chord accompaniment of unfamiliar melodies. The system was constructed under the subsumption architecture methodology. This paper discusses the design task of behavior decomposition for such a system, and recommends a strategy of modularizing the minimal adaptive requirements for the desired competence.

Keywords Music, subsumption architecture, behavior decomposition, adaptation, learning, neural networks

1 Introduction

Reactive, behavior-based systems have become widely used and accepted within some AI communities. A behavior-based system is one where control has been decomposed into a number of simpler and relatively autonomous *actions* that run concurrently. The intended overall behavior emerges from the continuous interaction of these components. In contrast, the traditional approach for software design is to decompose control into general *functions* and then construct the overall by stringing sequences of these behavior-segments together. If a behavior-based system is reactive, then each component behavior's output is a reflexive consequence of its inputs — there is no hidden complex computation or shared global state.

Although this new paradigm has its origins in general studies of mind and communication (see (Minsky 1986) or (Agre & Chapman 1988)) it has been most widely used and popularized as a method for controlling mobile robots. Where robots operating under the traditional control sequences had found the problem of moving across a dynamic free space almost intractable, robots using the new strategy can operate in such environments at speeds comparable to animals, using minimal computation (Brooks 1991 a). This success has been duplicated in the area of assembly robotics, where behavior-based control has been used to address the problems of sensor calibration and motion control, greatly simplifying the task of central assembly planning (Malcolm & Smithers 1990).

So far, however, the range of tasks done by fully reactive, behavior-based systems has been fairly limited, mostly to types of navigation and exploration. While the followers of the new practice are claiming to be establishing a new approach to artificial intelligence, skeptics have doubted whether "truly human" cognition can be achieved (Kirsh 1991). This question actually has at least two parts:

1. Can a reactive, behavior-based system perform tasks we consider to be acts of "higher cognition"?
2. If so, how do we know what constituent behaviors to create in order for an overall competence in these tasks to "emerge?"

As a beginning of an answer to both of these questions, this paper describes a recent research project, the Reactive Accompanist. The accompanist was developed under an approach called *subsumption architecture* as outlined in (Brooks 1991 b). Previously, subsumption architecture had been used exclusively for small, autonomous mobile robots — that is, robots with all their computation and power requirements carried on board. The RA carries the principles of this approach into a new domain — real-time musical expertise.

2 The Reactive Accompanist

The goal of the Reactive Accompanist is to derive chord structure from a melody in real time. This mimics the ability of skilled musicians to accompany unfamiliar melodies on chord instruments, such as the piano or the guitar. The input melodies are played live on traditional instruments, and the processing is totally reactive, with no transcription of the melody nor rules of harmonization applied. This research is most thoroughly described in (Bryson 1992). It was conducted as part of a Masters project for the Department of Artificial Intelligence at the University of Edinburgh in the summer of 1992. The Accompanist was written primarily in GNU C++ and implemented on a SUN SPARCstation SLC, using a the standard SUN sound equipment for audio input and output.

2.1 High Level Structure

The behaviors of a subsumption architecture program are organized into groups or *layers*. The layers each represent a level of competency. They are designed and implemented sequentially, beginning with the most basic. In the RA, we start with Pitch, which interprets microphone input as a pitch in a predetermined scale.[1] The next layer is Chord recognition, which transforms the output from pitches into chords. The following level is Time recognition.

[1] The RA was designed for use in any harmonic system, with the number of potential pitch classes set as a compile-time constant. However, it was only tested on the standard Western 12-tone chromatic scale.

Although the basic competence of this level is to recognize and anticipate the beats of its input melody, the output of the program at this level is not just rhythmic information, but the modified output of the chord layer. Now instead of a single chord, it outputs chords in a timed sequence.

SONG
STRUCTURE
TIME
CHORD
PITCH

Figure 1: Basic layers of a complete machine. Bold indicates implementation.

A proposed extension to the Accompanist includes a next layer that tries to recognize Structure in the piece. That is, it would bias chord choice towards recurring patterns[2]. The final level of competence would be Song recognition, whereby the program would learn to recognize song types. This would subsume Structure in its ability to anticipate the number of times a pattern would repeat, and what the next pattern is likely to be. Although these levels have not yet been broached, they are included here as further examples of high level decomposition.

2.2 Pitch

Pitch really consists of two behaviors, but one of them is currently external to the rest of the Accompanist software. That behavior is to translate microphone input into a set of frequency/gain pairs. Currently this is done via fast Fourier transformation using csound (Vercoe 1991). Unfortunately this relatively slow process moves the current implementation of the Accompanist out of real time, and necessitates the use of prerecorded SUN sound files.

The first behavior of the Accompanist proper is NOTE. This module is an artificial neural network. More specifically, it is a simple, fully connected, two-dimensional grid competitive net. The input lines correspond to ranges of frequencies, and the output nodes corresponding to the pitch classes. In order to bias the network to interpreting sound similarly to people, the ranges of the input lines are scaled logarithmically so that there are the same number of input lines between any two notes humans perceive as being octaves (that is, where the lower frequency is half the higher one). We chose on trial and error to have 96 frequency input lines per octave — we chose a low number for computational efficiency, but too few input lines lower the accuracy.

For our trials, this net was trained using supervision (a training file with known "correct" results). The training file consisted of slightly over four

[2]Rhythmic pattern bias is already implemented in the Time layer by the TIMED module.

minutes of a mandolin being plucked at the first twelve fret positions of each pair of strings. Which of these forty-eight intervals would be used was chosen at random, then the precise frame was chosen randomly from that interval. Note the frame could then consist of plucking noise, string noise, attack or any part of the note decay. Peak recognition tended to be about 86% correct, a figure that could partly reflect the noise of the file. The net trained up to 66% accuracy within 80 samples, and peaked by 200. After training, errors made often reflected dominant harmonics of a note.

If the NOTE net is trained without supervision, that is, the winning output node is reinforced regardless of accuracy, it will sometimes learn to distinguish multiple "pitches". These reflect actual pitch classes, but usually clustered unpredictably (neither necessarily harmonically nor sequentially) on just a few of the outputs. The fact that unsupervised training can reflect pitch rather than some other factor (such as plucking noise) probably indicates the utility of the initial frequency clustering bias.

The network trained on the mandolin file generalized well to penny whistle, Scottish small pipes, and four strings of an electric guitar. It failed on male and female human voices, a wood recorder, and the two (non-consecutive) other strings of the guitar.[3] The output of the network was taken to be the weights of *all* the output units, though in the training session only the winner was attended to. Using the full output array resulted in the Accompanist being able to exploit simple polyphonic (more than one note simultaneously) input for better chord choice.

2.3 Chord

The second layer is composed of only a single behavior, CHORD which is also a neural network. This is also a two-dimensional grid network, its design a hybrid of conventional associative and competitive networks. The input nodes correspond to each pitch class, with the input typically being an output or a summation of outputs from the NOTE module. The output nodes correspond to absolute chords, such as F minor or C major. The weights for each output node are referred to as a "chord template," and are algorithmically determined based on the notes actually present in the conventional definition of that chord. The output of any chord template that has received a single instance of a perfect match is 1, its output if it receives all pitch classes weighted evenly is 0. As with NOTE an array of the output values is preserved.

As an interesting incidental behavior, if an entire song is presented to the CHORD module as a single summed input, it will generally report the key of the song. This is a natural consequence of the dominance of the notes of the chord corresponding to that key in the song overall.

[3] We failed to find a pattern to what instruments this net could classify. All inputs were checked with conventional tuning devices. Microphone response may have been a factor.

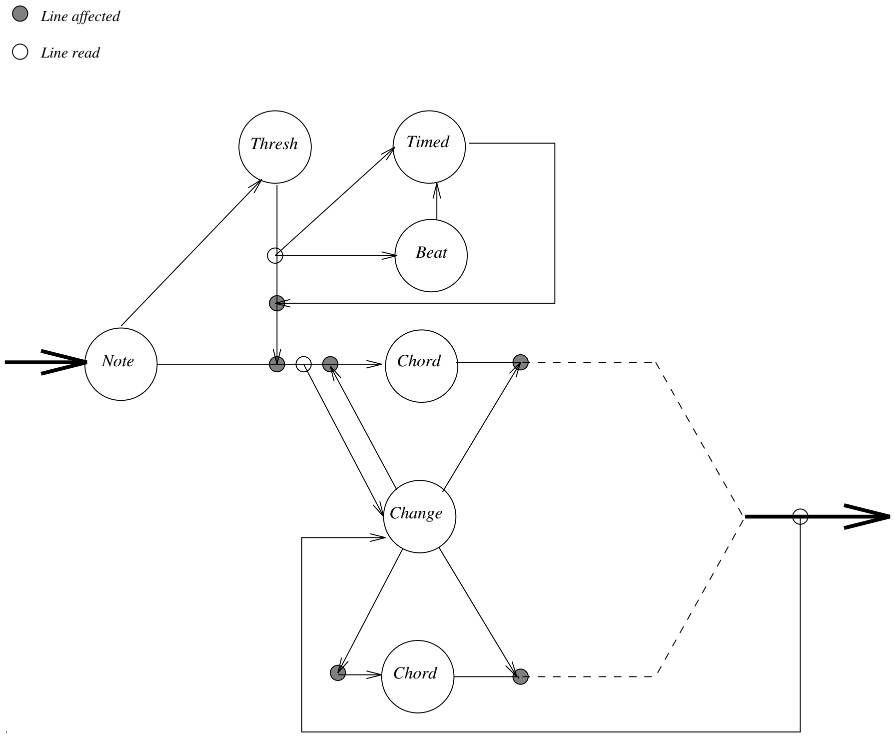

Figure 2: The interactions of the components of the Reactive Accompanist—circles represent modules, lines the flow of information

2.4 Time

The modules of the time layer all operate at the rate of the frame sampling of the Fourier transformation.[4] Note that were the Accompanist running in real time, it might be necessary to drop some frames to save processing time. All of the behaviors are by design robust with respect to this loss, providing the frame drops are distributed evenly (not clustered over harmonically significant events) and the rhythm modules are either aware of the *actual* frame number or else the frequency of frame droppings are consistent throughout a piece.

The time layer consists of four modules. The first three are responsible for determining the location of the beats within the melody. THRESH looks for the threshold of a note by noticing if a single pitch classification persists after a change. It consists of just a few register variables — the pitch and

[4] The experiments sampled with a frame size of 256 milliseconds and a frequency of 128 milliseconds.

the frame count. BEAT recognizes beats and pulses in the music using a strategy inspired by reported incidental results in (Desain & Honing 1992). It consists primarily of a one-dimensional neural network (just a vector array) with each (indexed) weight associated with a number of frames. It notes the intervals between positive results from THRESH and sums them (in a normal curve which includes neighboring weights) onto its weight array at that interval. Previous weights are decayed by the same amount as the new interval introduces; the overall sum of the weights is constant. Consequently over time the peak weight/index values can shift if the piece is changing tempo. Notice peaks will emerge at each common length for a note, thus pulses, beats and even measures may be found in the same array. TIMED outputs a true value at what it expects to be the threshold of a beat. It reacts to the peaks of BEAT in determining its interval and THRESH in determining its timing. Like THRESH it consists just of a few counters and register variables.

The other module of Time, CHANGE, is what introduces the possibility of chord change into the overall output of the accompanist. CHANGE maintains two competing copies of the CHORD behavior — feeding them both the input from NOTE. The output of whichever copy seems to be dominating is reported as the output of the whole module. The challenging CHORD could become dominant if the relative weight of its winning chord to the number of inputs seen is increasing while the dominant chord's is decreasing.[5] On a signal from TIMED of a new beat, CHANGE resets the value of the challenging chord. This makes a new chord more likely to emerge quickly, since it didn't need to compete with existing weights. (Chord changes are most likely to occur on the beat.) The correct new chord was found more quickly and accurately if the challenging chord was "primed" with weights that reflected seeing the current dominant (best guess) chord for a fraction of a time interval. This utility of context is due to the nature of traditional music — a following chord is likely to contain several of the notes of the previous one.

2.5 Results

On an input song on one of its preferred instruments, the Reactive Accompanist makes acceptable chord choices, on average, about 60% of the time. For simple songs or polyphonic input, performance can go up over 75%. Testing was somewhat arbitrary and qualitative, but was performed by several trained musicians. On examining the transcripts, the RA was found to make some of the same errors as beginning harmony students. It also had the entertaining if pathological behavior of refusing to change chords when accompanying the smallpipes, even when the drones [the continuous, single-tone pipes] were removed. This is probably a consequence of acoustic properties of the chanter. The accompanist did pick the correct drone chord.

[5] An alternate, simpler strategy of straight relative chord strength was found equivalent 80% of the time, but generally less accurate where there were discrepancies.

There are few comparable systems to the Reactive Accompanist — most intelligent systems for musical accompaniment follow scores. Those that improvise tend to take rhythmic and harmonic information as input and produce melodic lines. Nearly all systems work with MIDI (Musical Instrument Digital Interface) input rather than microphones. The RA was developed in just three and a half months, has unbiased access to 24 chords (major and minor of all 12 pitch classes), and produces output that people perceive as intelligent and interesting.

3 Adaptation and Behavior Decomposition

Having presented an already functional decomposition, the question becomes "how did you know to do it that way?". The answer for this project is largely the same as it would be for any of the traditional systems we construct — thought, intuition, experience in programming [6], and a certain amount of trial and error. However, we can trace some of the fundamental decisions involved.

The first step was to divide the task into Brooks' "levels of competence" (Brooks 1991 b). It is widely accepted that pitch and time are the fundamental components of music, but the decision to concentrate first on pitch was more arbitrary. The primary motivation there was that the ability to recognize pitch and harmony seemed more fundamental and easily separable from the overall task than the ability to recognize rhythmic events.

What is probably more interesting is the nature and number of the behavior modules within the levels. In the RA, why are some of the modules neural networks, while others are not? Why are there several different neural networks, for example the chord and note competences, instead of one large general network? Again, the original reason can be put down to intuition. But we suspect the reason this approach proved successful is because it divides the problem along the lines of when and how behavior needed to be adapted. Some information, such as what nodes compose a chord, is completely external to the project, and thus behaviors dependent on that information need never adapt. Other information, such as what notes are being played *right now*, is completely dynamic and must be learned in real time. Before a more complete discussion of these divisions in the Accompanist, we first define these different forms of learning.

[6] Although this was my first experience in subsumption architecture, my knowledge of traditional programming algorithms certainly gave me some guidance as to what could and could not be expected to be achieved in translating a set of inputs into a desired set of outputs.

3.1 Three Types of Learning

First, it must be acknowledged that the use of the term "learning" is dangerous, since many researchers associate the word with specific kinds of processes. Within the context of this paper, *learning* means any process which equips an agent with a new behavior which it can perform autonomously. This definition is deliberately broad enough to include such things as hardware design.

Under this definition, forms of learning can be broken into three classes:

- provided learning,
- required learning, and
- open learning.

Provided learning is the behaviors determined without any action by the intelligent agent itself. For animals, this includes the bodies they are born with, their reflexes, and some extent of what might be called "instinctive behavior". For robots, this is their physical hardware, and any software behaviors that have no variable storage content, that is, that are not influenced by the agent's own previous experience but only by its direct current environment. In the RA, an example is the chord templates.

Required learning is domain-specific, preordained learning performed by the individual agent that is required for its normal "adult" behavior. Examples from biology include things like the learning of stellar maps and songs in birds, or bees learning the ephemeris of the sun for navigation. In robotics, examples include self-calibration of sensors or preliminary training of recognition networks, like the Accompanist's NOTE net. This kind of learning can often be thought of in terms of parameter adjustment. Often it occurs in animals during specific periods of their early development, while in robots it might occur before the the robot is considered fully functional. Some "recalibration" sorts of required learning persist through the whole lifetime of the agent.

Open learning is the closest equivalent to the traditional definition of learning — that of a general associative mechanism that allows us to classify and respond to all our surroundings. It is freer and more general than required learning, and is likely to occur over the entire lifetime of the agent. Examples of this are basic conditioning in animals and the ability to "compile" action sequences into expert skills in humans.

Open learning is still not necessarily the kind of complete, general, logical learning long proposed as the whole story by traditional psychology. Not only does open learning coexist with, and in most cases play a subordinate role to, provided or required learning, but it may well be constrained to specific domains. For an example from classical conditioning, pigeons can learn to peck for food, but cannot learn to peck to avoid a shock. They can, however, learn to flap their wings to avoid a shock (Hineline & Rachlin 1969). This research is cited in (Gallistel, Brown, Carey, Gelman & Keil 1991) as evidence that most dynamic learning in animals tends to be specialized and constrained, *not* generalized. Gallistel et al. (1991) in fact suggests that nature only uses individual adaptation as a last resort, for behaviors that cannot be provided on an evolutionary time scale. For example, the stellar maps that some migratory birds imprint on in the nest change too fast for natural selection to keep up.

We introduce these classifications of learning largely to understand and to reference the distinctions in animal learning brought out in (Gallistel et al. 1991)'s survey. Particularly in biology, the border-lines between classifications may be unclear in some cases; for example, the physical growth of the organism. In an artifact, however, these distinctions are clear because one can easily distinguish the mechanisms that underlie them. Provided learning will never change over the artifact's lifetime. Required learning adjusts specific parameters in narrow, well defined ways, using special purpose mechanisms. Open learning uses general techniques to add new skills, new reactions. Unfortunately, the organic ambiguity is reintroduced in the engineers' decision of which behaviors should fall into which class of learning. Our recommendation is to follow the hypothesis of Gallistel et al. (1991), that learning should be no more general than necessary.

3.2 The Reactive Accompanist in Terms of Modularized Adaptivity

The Reactive Accompanist needs to adapt to environmental factors along many axes. For example:

- The number and intervals of *pitch classes* in a scale.
- The number and description of *chord templates* in a harmonic system.
- The *frequency signature* a particular instrument makes in creating a tone.
- The *note* that is currently being played.
- The occurrence of a *pitch change*.
- The *tempo* at which the piece is currently being played.
- The number of different notes that might be played within a given beat (*pulses*).

Nothing in the Accompanist actually reacts *to* a chord or a chord change in the melody. Rather, the *apparent interpretation* of a melody into a sequence of chords emerges from the behavior of the constituent parts.

Pitch classes and chord templates don't vary at all within the scope of an execution for the Accompanist, even if that execution were to encompass several songs. Thus they are provided learning. The frequency signature might also be, though adapting to different instruments would be useful since there is not complete generalization. The signature learning is required instead of provided more because of its complexity – it is not practical to provide it in a reasonable time frame.

The remaining variables must be adapted to ("learned" as defined by (Gallistel et al. 1991)) in real time. They are each handled in a highly specialized provided learning mechanism segregated into a single behavior. Note recognition takes place in NOTE, Pitch change in THRESH, tempo and pulses in BEAT.

Other behaviors that are not dealing directly with adaptation to external events are dealing with integrating the results of other modules into useful form. CHORD connects note and the provided chord templates, TIMED integrates beat and thresh for CHANGE, and CHANGE integrates the rhythm information with CHORD.

Notice the Reactive Accompanist has no open learning mechanisms at all. Few subsumption-architecture projects do, but this is not out of necessity. Mataric (1990) uses a highly specialized open learning module for map learning in navigation.[7] Open learning could have obvious uses in higher levels of control programs designed to learn experientially. Probably the reason it has been avoided thus far is that open learning is normally conceived of only in terms of standard "general learning" with its requisite symbolic-logic structure. Implementing such a system would violate one of the basic tenets of subsumption architecture — no representation (Brooks 1991 b). The positive result of this restriction has been very fast, robust behavior in the reactive, behavior-based systems. The negative side has been complaints that the systems don't do enough "interesting" things. This complaint is not only addressed in the RA, but also in other current research (see below). Regardless, the subsumption architecture approach to control has proven naturally conducive to following the strategy of specializing learning as much as possible.

4 Current Research

There is much more work to be done on the Reactive Accompanist. As it stands, the RA has little sense of context on the level of musical phrase, let alone song. Consequently, it can make what to humans seem like obvious mistakes, such as failing to chose the appropriate final chord to match the single held tone that typically ends a phrase of folk music. Also, from a technological standpoint, there are several new strategies available for conducting the frequency analysis necessary for the input to the initial network that could move the project truly into real time. Hopefully these expansions will be made within the next few years.

Currently we are involved in a more ambitious project of human-level cognition being conducted at MIT. This research, referred to as "the humanoid project" is a major large-team effort to replicate many aspects of human behavior. In concentrating on human infants as a model, the humanoid project

[7](Bryson 1992) suggests using a similar system to enable higher levels of the Reactive Accompanist to "map" songs and musical phrases.

is provided with a well-researched natural decomposition and ordering of levels of competence. We will be testing many models of learning and adaptation on this project, but acquisition of musical skills is still some time off in the future.

5 Conclusions

Both reactive and behavior-based models of cognition are still very young, even for a field like artificial intelligence. Nevertheless, they are proving fertile, not only in providing engineering solutions, but also to research in ethology and philosophy in mind. Projects like the Reactive Accompanist can help us explore both the new techniques of engineering and the potentials and limitations of the new cognitive models.

In the Reactive Accompanist, we have demonstrated an implementation of a human competence well outside the conventional domain of strict physical reflexive behavior some have suggested is the limit of the utility of subsumption-architecture styled systems. The Accompanist shows a musical competence human listeners consider to sound intelligent, if not particularly gifted.

The Reactive Accompanist also serves as a new example of behavior decomposition under subsumption architecture. This paper has introduced a dichotomy of behavior acquisition — into provided, required, and open learning — as a framework for analyzing tasks. We support the strategy of limiting and specializing adaptability as much as possible. We suggest compartmentalizing the different learning competencies as both a development methodology for and one of the strengths of behavior-based systems.

Acknowledgments

Several people have urged me to write a paper on the decomposition problem, most notably my supervisors on the Reactive Accompanist project, Alan Smaill and Geraint Wiggins, and also Morten Christiansen who suggested some of the direction of this paper. The learning and ethology research was conducted at the suggestion of Lynn Stein and Rodney Brooks. Thanks to Bill Smart for his contributions and proofreading. Besides myself, musicians for the Reactive Accompanist were Manuel Trucco, Sheila Tuli, Colin Macnee and Helen Lowe.

References

Agre, P. E. & Chapman, D. (1988), What are plans for?, AI memo 1050, MIT, Cambridge, MA

Brooks, R. A. (1991 a), Intelligence without reason, in Proceedings of the 1991 International Joint Conference on Artificial Intelligence, pp. 569–595

Brooks, R. A. (1991 b), Intelligence without representation, Artificial Intelligence 47(1–3), 139–160

Bryson, J. (1992), The subsumption strategy development of a music modelling system, Master's thesis, Department of Artificial Intelligence, University of Edinburgh

Desain, P. & Honing, H. (1992), Music, Mind and Machine: Studies in Computer Music, Music Cognition, and Artificial Intelligence, Thesis Publishers, Amsterdam

Gallistel, C., Brown, A. L., Carey, S., Gelman, R. & Keil, F. C. (1991), Lessons from animal learning for the study of cognitive development, in S. Carey & R. Gelman (eds.) The Epigenesis of Mind, Lawrence Erlbaum, Hillsdale, NJ

Hineline, P. & Rachlin, H. (1969), Escape and avoidance of shock by pigeons pecking a key, Journal of Experimental Analysis of Behavior 12, 533–538

Kirsh, D. (1991), Today the earwig, tomorrow man?, Artificial Intelligence 47, 161–184

Malcolm, C. & Smithers, T. (1990), Symbol grounding via a hybrid architecture in an autonomous assembly system, in P. Maes (ed.) Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, MIT Press, Cambridge, MA, pp. 123–144

Mataric, M. J. (1990), A distributed model for mobile robot environment-learning and navigation, AI memo 1228, MIT, Cambridge, MA

Minsky, M. (1986), The Society of Mind, Simon and Schuster, New York, NY

Vercoe, B. (1991), The Csound Reference Manual, Cambridge, MA